Homework1-CreditDataset

Sophia Oku

2023-01-14

```
#getwd()

#install.packages("ISLR") - not necessary
#install.packages("e1071") - not necessary
#install.packages("catools") - not necessary
#install.packages("kernlab")

df <- read.table(file = "credit_card_data-headers.txt", header = TRUE)
str(df)

## 'data.frame':    654 obs. of  11 variables:
##  $ A1 : int  1 0 0 1 1 1 1 0 1 1 ...
##  $ A2 : num  30.8 58.7 24.5 27.8 20.2 ...
##  $ A3 : num  0 4.46 0.5 1.54 5.62 ...
##  $ A8 : num  1.25 3.04 1.5 3.75 1.71 ...
##  $ A9 : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ A10: int  0 0 1 0 1 1 1 1 1 1 ...
##  $ A11: int  1 6 0 5 0 0 0 0 0 0 ...
##  $ A12: int  1 1 1 0 1 0 0 1 1 0 ...
##  $ A14: int  202 43 280 100 120 360 164 80 180 52 ...
##  $ A15: int  0 560 824 3 0 0 31285 1349 314 1442 ...
##  $ R1 : int  1 1 1 1 1 1 1 1 1 1 ...

head(df)

##   A1    A2    A3   A8 A9 A10 A11 A12 A14 A15 R1
## 1  1 30.83 0.000 1.25  1   0   1   1 202   0  1
## 2  0 58.67 4.460 3.04  1   0   6   1  43 560  1
## 3  0 24.50 0.500 1.50  1   1   0   1 280 824  1
## 4  1 27.83 1.540 3.75  1   0   5   0 100   3  1
## 5  1 20.17 5.625 1.71  1   1   0   1 120   0  1
## 6  1 32.08 4.000 2.50  1   1   0   0 360   0  1

#sum(is.na(data)) - read the notes and realized the NA's have been removed.

library(kernlab)
model <- ksvm(as.matrix(df[,1:10]),as.factor(df[,11]),type="C-
svc",kernel="vanilladot",C=100,scaled=TRUE)

##  Setting default kernel parameters

model
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 100
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 189
##
## Objective Function Value : -17887.92
## Training error : 0.136086
```

```r
#Training error for model : 0.136086

model2 <- ksvm(as.matrix(df[,1:10]),as.factor(df[,11]),type="C-svc",kernel="vanilladot",C=150,scaled=TRUE)
```

```
##  Setting default kernel parameters
```

```r
model2
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 150
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 193
##
## Objective Function Value : -26831.5
## Training error : 0.136086
```

```r
model3 <- model2 <- ksvm(as.matrix(df[,1:10]),as.factor(df[,11]),type="C-svc",kernel="vanilladot",C=50,scaled=TRUE)
```

```
##  Setting default kernel parameters
```

```r
model3
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 50
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 189
##
## Objective Function Value : -8944.221
## Training error : 0.136086
```

```
#Training error : 0.136086  remains the same for model 1- 3.

# calculate a1…am
a <- colSums(model@xmatrix[[1]] * model@coef[[1]])
print(a)

##            A1            A2            A3            A8            A9
## -0.0010065348 -0.0011729048 -0.0016261967  0.0030064203  1.0049405641
##           A10           A11           A12           A14           A15
## -0.0028259432  0.0002600295 -0.0005349551 -0.0012283758  0.1063633995

# calculate a0
a0 <- -model@b
print(a0)

## [1] 0.08158492

# see what the model predicts
pred <- predict(model,df[,1:10])
pred

##   [1] 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
##  [38] 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1
1 1 0
##  [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [112] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [149] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [186] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 1
## [223] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1
## [260] 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0
## [297] 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0
## [334] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
## [371] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
## [408] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0
## [445] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [482] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 1 1
## [519] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1 0 1
```

```
## [556] 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0
## [593] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0
0 0 0
## [630] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## Levels: 0 1
```

```
sum(pred == df[,11]) / nrow(df)
```

```
## [1] 0.8639144
```

```
#2) using rbfdot and anovado
#?ksvm
model4 <- ksvm(as.matrix(df[,1:10]),as.factor(df[,11]),type="C-
svc",kernel="anovado",C=50,scaled=TRUE)
```

```
##  Setting default kernel parameters
```

```
model4
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 50
##
## Anova RBF kernel function.
##  Hyperparameter : sigma =  1 degree =  1
##
## Number of Support Vectors : 201
##
## Objective Function Value : -8320.227
## Training error : 0.11315
```

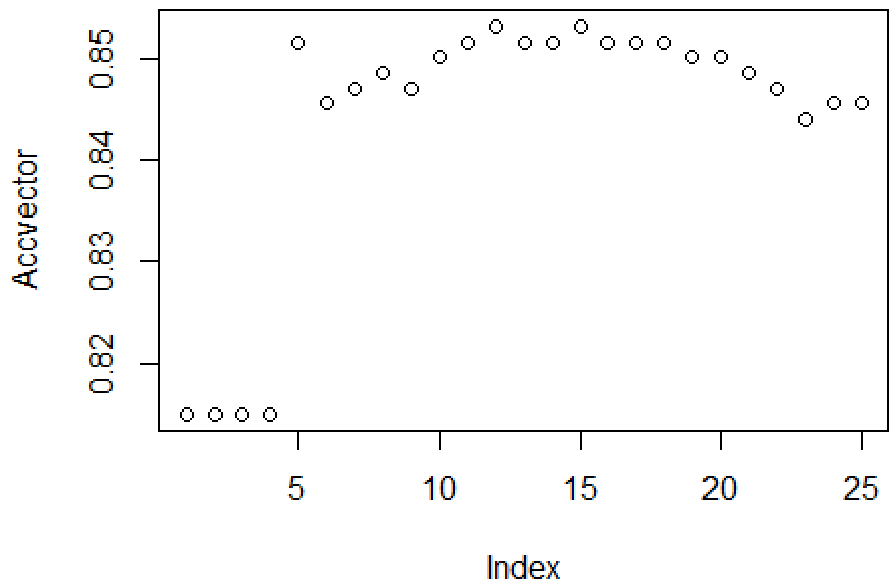
```
model5 <- ksvm(as.matrix(df[,1:10]),as.factor(df[,11]),type="C-
svc",kernel="rbfdot",C=50,scaled=TRUE)
model5
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: C-svc  (classification)
##  parameter : cost C = 50
##
## Gaussian Radial Basis kernel function.
##  Hyperparameter : sigma =  0.0949687047382442
##
## Number of Support Vectors : 245
##
## Objective Function Value : -5280.63
## Training error : 0.056575
```

```
#model4 using the Anovado kernel has Training error : 0.11315
#model5 using the rbfdot kernel  has Training error : 0.062691
```

```
#model using the vanilladot kernel has Training error : 0.136086

#since we are looking to minimize the training error, model5 will produce the
best outcome.

#to calculate the coefficients
a4 <- colSums(model4@xmatrix[[1]] * model4@coef[[1]])
a5 <- colSums(model5@xmatrix[[1]] * model5@coef[[1]])

a4

##           A1           A2           A3           A8           A9
A10
##  -0.01627274 -15.30650271 -21.73660856   1.59902355   2.43833685  -
0.77742131
##          A11          A12          A14          A15
##   2.40204228  -0.02702823 -12.69067598  18.06352121

a5

##         A1         A2         A3         A8         A9        A10
A11
##  -8.761213 -33.763965  -4.454269  48.762358  37.970436 -15.022432
11.206703
##        A12        A14        A15
## -17.688853 -49.305542  43.590095

a40 <- -model4@b
a50 <- -model5@b

a40

## [1] 0.9346853

a50

## [1] 0.730902

# see what the model predicts
pred4 <- predict(model4,df[,1:10])

pred5 <- predict(model5,df[,1:10])

sum(pred4 == df[,11]) / nrow(df)

## [1] 0.8868502

sum(pred5 == df[,11]) / nrow(df)

## [1] 0.9434251
```

```r
#as expected model5 (rbfdot) produced higher prediction correlation in
respect to
#actual classification
#0.9373089 - 94%

library(kknn)
Accvector <- c()
PredictionKKnn <- c()


# Iterating in a loop for 25 possible k values
for (K in 1:25) {
  # Use lapply to train a model to run for each value of k indicated as "z"
  kknn_mod1 <- lapply(1:nrow(df), function(z) {
    kknn_mod2 <- kknn(df[-z,11]~., df[-z,1:10],df[z,1:11],k = K,kernel =
"optimal",scale = TRUE)
    return(kknn_mod2)
  })
  # Using lapply to make predictions for each model and rounded
  PredictionKKnn <- lapply(kknn_mod1, function(x) round(fitted(x)))
  # Flatten the predictions to produce a vector
  PredictionKKnn <- unlist(PredictionKKnn)
  # Calculating the accuracy
  KnnAccy <- sum(PredictionKKnn == df[,11]) / nrow(df)
  #combining the accuracy for each K into the blank vector
  Accvector <- c(Accvector,KnnAccy)
}

plot(Accvector)
```




```
#install.packages("caret")
library(caret)

## Loading required package: ggplot2

##
## Attaching package: 'ggplot2'

## The following object is masked from 'package:kernlab':
##
##     alpha

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:kknn':
##
##     contr.dummy

#confusion matrix

class(df$R1) #numeric

## [1] "integer"

class(PredictionKKnn) #integer
```

```
## [1] "numeric"
```

```
confusionMatrix(as.factor(df$R1),as.factor(PredictionKKnn))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 302  56
##          1  45 251
##
##                Accuracy : 0.8456
##                  95% CI : (0.8156, 0.8724)
##     No Information Rate : 0.5306
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.6893
##
##  Mcnemar's Test P-Value : 0.3197
##
##             Sensitivity : 0.8703
##             Specificity : 0.8176
##          Pos Pred Value : 0.8436
##          Neg Pred Value : 0.8480
##              Prevalence : 0.5306
##          Detection Rate : 0.4618
##    Detection Prevalence : 0.5474
##       Balanced Accuracy : 0.8440
##
##        'Positive' Class : 0
##
```

```
#accuracy of the model using the optimal scale is @ 84.5%

which.max(Accvector)
```

```
## [1] 12
```

```
which.min(Accvector)
```

```
## [1] 1
```